

Improved Lower Bound on DHP: Towards the Equivalence of DHP and DLP for Important Elliptic Curves Used for Implementation

Prabhat Kushwaha
IISER Pune, Dr. Homi Bhabha Road, Pashan
Pune-411008, India
prabhat1412@gmail.com

November 29, 2016

Abstract

In 2004, Muzereau *et al.* showed how to use a reduction algorithm of the discrete logarithm problem to Diffie-Hellman problem in order to estimate lower bound on Diffie-Hellman problem on elliptic curves. They presented their estimates for various elliptic curves that are used in practical applications. In this paper, we show that a much tighter lower bound for Diffie-Hellman problem on those curves can be achieved, if one uses the multiplicative group of a finite field as an auxiliary group. Moreover, improved lower bound estimates on Diffie-Hellman problem for various recommended curves are also given which are the **tightest**; thus, leading us towards the equivalence of Diffie-Hellman problem and the discrete logarithm problem for these recommended elliptic curves.

Keywords: Discrete Logarithm problem, lower bound of the Diffie-Hellman problem, elliptic curves used in practical applications.

1 Introduction

It is well known that the discrete logarithm problem(DLP) is one of two primitives that are commonly used as a building block in public key protocols, other being integer factorization. Computational difficulty in solving DLP is a security necessity for the protocols based on it. However, interesting thing about these DLP-based protocols is that, security of many of such protocols does not exactly rely on the hardness of DLP. For example, the ElGamal public key cryptosystem is secure if and only if the Diffie-Hellman problem(DHP) is hard to solve [4, Proposition 2.10]. That means, it is enough for an attacker to solve DHP to break the ElGamal cryptosystem. The Diffie-Hellman key exchange, pairing-based cryptosystems, digital signature schemes and many more protocols are some other examples where the security of the protocol depends on hardness on DHP. This is why hardness of DHP is of utmost importance in public key cryptography.

If DLP is easy, DHP is easy because a solution of DLP immediately yields a solution of DHP. Therefore, the only meaningful scenario to study the hardness of DHP is when DLP is known to be hard. Barring some weak elliptic curves over finite fields, there are no efficient algorithm to solve the discrete logarithm problem on the group of points of an elliptic curve over finite field (ECDLP) and thus, those elliptic curves are widely used for practical purposes. Thus, it is of paramount importance to study about the hardness of the elliptic curve Diffie-Hellman problem (ECDHP) from the point of view of practical cryptography. The **central theme** of this paper is to study the hardness of ECDHP since a number of public key protocols are designed on such curves and their security depends only on the hardness of ECDHP.

1.1 Summary of Existing work

To study the hardness of DHP the traditional method, and the only method known so far, involves reduction arguments from DLP to DHP. In this section, we will summarize those reduction arguments. In such reductions, one tries to solve DLP efficiently (in polynomial time of the input bit), using the existence of a solution of DHP as sub-routine. If there exists such an algorithm, we say that **DLP reduces to DHP** in polynomial time, denoted by $\text{DLP} \leq_P \text{DHP}$. Informally, if $\text{DLP} \leq_P \text{DHP}$, it implies that DHP is at least as hard as DLP, or equivalently, DLP is no harder than DHP. Clearly, existence of any such reduction algorithm in case of elliptic curve groups would imply that ECDHP is hard, since ECDLP is hard to solve.

The first, and the only, reduction algorithm known so far which reduces DLP to DHP was proposed by Maurer in his seminal paper [6]. He introduced the technique of implicit representation of elements of a finite field and indicated the use of an auxiliary group in constructing such a reduction algorithm. Soon after that, Maurer and Wolf showed that $\text{DLP} \leq_P \text{DHP}$ for any group \mathbb{G} of prime order p ; if we are able to find an elliptic curve over \mathbb{F}_p with smooth order [7–10]. Smooth order of the auxiliary elliptic curve was the main reason behind the polynomial time reduction of DLP to DHP in their algorithm because it ensures that the total number of group operations as well as the total number of calls to the DH-oracle required in their algorithm remain polynomial in the input size. However, it is exceptionally hard, in general, to construct an elliptic curve over \mathbb{F}_p of smooth order for large p , resulting in the failure of above theorem. Therefore, some alternate method was needed to study the hardness of DHP.

In 2004, Muzereau *et al* re-visited the Maurer and Wolf reduction algorithm of DLP to DHP for special case of elliptic curve groups over finite field [12]. They explicitly constructed auxiliary elliptic curves, which were required in the reduction algorithm, for a number of elliptic curves recommended for practical implementation in *SEC 2* [14] by Standard for Efficient Cryptography Group (SECG) at Certicom Corporation. We will refer to those recommended elliptic curves in *SEC 2* as **SECG curves** [14] throughout this paper. However, the orders of the auxiliary elliptic curves constructed were not smooth enough, making the cost of the reduction algorithm exponential. Therefore, their reduction algorithm with those auxiliary elliptic curves failed to prove

the polynomial reduction of ECDLP to ECDHP for those recommended SECG curves. Nevertheless, all was not lost as it might seem, since they were the first to give precise estimate of the number of group operations needed in such a reduction algorithm and showed how to use such a reduction algorithm to estimate the minimum number of group operations required to solve ECDHP on those SECG curves [12, Table 1, Table 2].

Bentahar later applied the idea similar to Muzereau *et al.* but constructed different auxiliary elliptic curves over \mathbb{F}_p to improve those estimates of Muzereau *et al.* on the exact lower bounds on ECDHP for those important SECG curves [2, Table 1, Table 2]. Since those lower bounds on ECDHP are assumed to be beyond the reach of present computational power, it establishes the security of several protocols relying on the hardness of ECDHP for those recommended SECG curves. This shows the significance of this remarkable approach.

1.2 Our Contribution

The algorithms of Muzereau and Bentahar both use the same reduction algorithm suggested by Maurer and Wolf. They used suitable elliptic curves over a finite field as auxiliary groups. Our contribution in this paper is that a new reduction algorithm of DLP to DHP is presented which uses, *for the first time*, the multiplicative group of a finite field as an auxiliary group. Our reduction algorithm is also different from those used by Muzereau [12] and Bentahar [2] or from any other previous reduction algorithm. Owing to this difference between our algorithm and previous algorithms and the change in the auxiliary group from an elliptic curve over a finite field to the multiplicative group of a finite field; our reduction algorithm requires very small number of DH-oracle calls. Our reduction algorithm results in increasing the lower bound on DHP, because the lower bound on DHP is inversely proportional to the number of calls to the DH-oracle. When applied to SECG curves studied first by Muzereau *et al.* and then by Bentahar, our reduction algorithm improves the previous lower bounds on ECDHP.

More precisely, assuming that the best algorithm to solve DLP on an elliptic curve of order p takes at least \sqrt{p} group operations, Muzereau *et al.* gave the following estimate on the lower bound on ECDHP [12, Theorem 4]:

Theorem 1. *Let p be a prime. Assuming in the interval $[p+1-\sqrt{p}, p+1+\sqrt{p}]$ there is an integer which is product of three primes of roughly equal size, then there exists a string S which implies that the best algorithm to solve the ECDHP for an elliptic curve of order p takes time at least*

$$O\left(\frac{\sqrt{p}}{(\log_2 p)^2}\right)$$

group operations.

Under the same assumptions as above, our reduction algorithm will prove the following theorem which improves the lower bound on ECDHP in Theorem 1:

Theorem 2. *For a prime p , assume that there exists a divisor d of $p-1$ of the size roughly equal to $\sqrt[3]{p}$. Then, the best algorithm to solve ECDHP for an*

elliptic curve of order p takes at least

$$O\left(\frac{\sqrt{p}}{\log_2 d}\right)$$

group operations.

It is important to note that both the theorems above assume that the best known algorithm to solve ECDLP on an elliptic curve of order p requires at least \sqrt{p} group operations.

Our result is significant as it applies to almost all the recommended SECG curves because such a divisor d exists for almost all of those curves where prime p is either the order of those elliptic curve groups or the largest prime divisor (with a very small co-factor of either 2 or 4).

Moreover, for curves SECP521R1, SECT409R1, SECT571R1, SECT571K1, Bentahar was unable to construct the auxiliary elliptic curves. However, we had no problem applying our algorithm to these curves and the lower bound estimates on these curves are also given here.

2 Notations and Definitions

Let $\langle \mathbb{G}, + \rangle$ be a cyclic (additive) group generated by P and order of P is a prime p .

Definition 1. Given $Q \in \mathbb{G}$, the problem of computing the integer x modulo p such that $Q = xP$ is called the **discrete logarithm problem (DLP)** with respect to P .

Definition 2. Given $Q = xP, R = yP \in \mathbb{G}$ (x, y are unknown integers), the problem of computing $S = xyP$ is called the **Diffie-Hellman problem (DHP)** with respect to P .

From the above definitions, it is clear that if one can compute x from $Q = xP$ and then he can compute $xR = xyP \in \mathbb{G}$. Thus, the solution of DLP readily yields the solution of DHP. However, as discussed earlier, we are interested in the reverse implication: does a solution of DHP solve DLP as well? To answer this question, reduction of DLP to DHP has been suggested and also given in some particular cases by Maurer and Wolf. As mentioned earlier, one tries to solve DLP assuming that a solution of DHP is known, or equivalently, one has access to a DH-oracle. We define it formally as follows:

Definition 3. A **DH-oracle** is a function that takes $xP, yP \in \mathbb{G}$ as inputs and returns $xyP \in \mathbb{G}$ as output. We write it as $\mathcal{DH}(xP, yP) = xyP$.

It was great insight of Maurer and Wolf who gave the first reduction algorithm that solved DLP using the DH-oracle as a sub-routine. The algorithm used the idea of *implicit representation of elements of a finite field \mathbb{F}_p and auxiliary groups*.

2.1 Implicit Representation of Elements of \mathbb{F}_p

Let \mathbb{G} be a cyclic group with generator P whose order is a prime number p . Let $y \in \mathbb{F}_p$. Then, $yP \in \mathbb{G}$ is called the *implicit representation of $y \in \mathbb{F}_p$ (with respect to \mathbb{G} and P)*. We denote this by $y \rightsquigarrow yP$.

Let $yP, zP \in \mathbb{G}$ be implicit representations of $y, z \in \mathbb{F}_p$ respectively. Then following basic algebraic operations in \mathbb{F}_p can also be realized in \mathbb{G} as follows:

- **Equality testing:** $y = z$ if and only if $yP = zP$.
- **Addition:** $y + z \rightsquigarrow yP + zP$ (1 group operation in \mathbb{G}).
- **Subtraction:** $y - z \rightsquigarrow yP - zP$ ($O(\log p)$ group operations in \mathbb{G}).
- **Multiplication:** $y \cdot z \rightsquigarrow yzP = \mathcal{DH}(yP, zP)$ (1 call to DH-oracle).
- **Inversion:** $y^{-1} = y^{p-2} = \underbrace{y \cdots y}_{p-2 \text{ times}} \rightsquigarrow y^{p-2}P$ ($O(\log_2 p)$ DH-oracle calls by using binary expansion).

Observe that the DH-oracle is used only for multiplication and inversion in \mathbb{F}_p . Therefore, number of DH-oracle calls required in the reduction algorithm increases with the increase in number of multiplication and inversions in \mathbb{F}_p required in the reduction algorithm. We will see the importance of this in later sections.

2.2 Auxiliary Groups

As the name suggests, any group (other than the group \mathbb{G}) is called an auxiliary group if it can be used to achieve the targeted goal of an algorithm. In the present context of DLP to DHP reduction, the goal is to solve DLP using the DH-oracle calls and implicit representations. Therefore, two essential properties of a possible auxiliary group \mathbb{H} are:

- Elements of \mathbb{H} can be represented as m -tuples of elements of \mathbb{F}_p for some $m \geq 1$.
- Group operation in this auxiliary group \mathbb{H} can be defined from algebraic operations in \mathbb{F}_p .

These two necessary properties of \mathbb{H} were suggested by Maurer and Wolf [8]. If \mathbb{H} has these properties, then any computation in \mathbb{H} (for example, equality testing, exponentiation in \mathbb{H}) can also be performed on their implicitly represented elements of \mathbb{G} . For more details, refer to [8].

Moreover, Maurer and Wolf [8] also mentioned two classes of possible auxiliary groups, satisfying above requirements: elliptic curves $\bar{E}(\mathbb{F}_p)$ and subgroups of $\mathbb{F}_{p^n}^\times$ for some $n \geq 1$. They called these groups *applicable auxiliary groups over \mathbb{F}_p* .

2.3 General Idea of Solving DLP using Auxiliary Groups and Implicit Representation Computation

Let \mathbb{G} be a cyclic group generated by P and order of P is a prime p . To solve DLP for Q , one requires to find the integer x where $Q = xP$. Moreover, we also have access to a DH-oracle on \mathbb{G} and we are allowed to make calls to the DH-oracle to solve DLP on \mathbb{G} . To this end, using auxiliary groups over \mathbb{F}_p and computation on implicitly represented elements of \mathbb{F}_p , Maurer and Wolf gave the following general idea for a DLP to DHP reduction algorithm:

1. Choose a cyclic auxiliary group \mathbb{H} over \mathbb{F}_p generated by ζ_0 .
2. **Embed** the unknown x into an implicitly represented element c of \mathbb{H} .
3. Compute discrete logarithm of c with respect to ζ_0 in \mathbb{H} explicitly, using computation(in \mathbb{G}) of implicitly represented elements of \mathbb{F}_p . Observe that computing implicit representations of finite field elements is exactly the place where the DH-oracle is used.
4. **Extract** the unknown x from the discrete logarithm of c with respect to ζ_0 found in the last step.

It is interesting to note that all DLP to DHP reduction algorithms known so far are based on Maurer and Wolf's idea of implicit representations. More intriguing is the fact that as auxiliary groups, only elliptic curves over \mathbb{F}_p of smooth order have been used and studied extensively.

If we take $\mathbb{H} = \bar{E}(\mathbb{F}_p)$ as the auxiliary group with smooth order N where elliptic curve $\bar{E}(\mathbb{F}_p)$ is given by $Y^2 = X^3 + AX + B$; $A, B \in \mathbb{F}_p$ and generated by $P_0 = (x_0, y_0) \in \mathbb{H}$, the reduction algorithm of Muzereau *et al.* (which follows the above general idea) **embeds** the unknown x implicitly into $c = Q_0 = (x, y) \in \mathbb{H}$ for some $y \in \mathbb{F}_p$. After that, the discrete logarithm k of Q_0 with respect to P_0 is computed *explicitly* using computations on implicitly represented elements. The last step is to **extract** x from $kP_0 (= Q_0)$ which is the abscissa of the point Q_0 . Observe that $Q_0 = (x, y)$ was not *explicitly* known before the computation of k . However, once we have k , we can compute Q_0 *explicitly* using P_0 and k as $Q_0 = kP_0$. Muzereau *et al.* first computed k modulo each prime power of N by repeatedly applying Pohlig-Hellman algorithm on implicitly represented elements along with exhaustive search to find a collision, then used the Chinese Remainder Theorem to find k . Bentahar also applied the same method. We call this several instances of Pohlig-Hellman algorithm and exhaustive search in their reduction algorithms collectively as **sub-algorithm A**. For more details on this reduction, see [12].

In the following section, we present a reduction algorithm that uses \mathbb{F}_p^\times as an auxiliary group, instead of an elliptic curve over \mathbb{F}_p .

3 DLP to DHP Reduction Algorithm using \mathbb{F}_p^\times as an Auxiliary Group

The reduction algorithm presented here is an adaptation of Cheon's work used to solve DLPwAI [3, Theorem 1]. Cheon analyzed the security concerns on DLP given some additional(auxiliary) input. Much to our surprise, we found that his algorithm fits perfectly well into the general idea of Maurer and Wolf to reduce DLP to DHP using implicit representation with \mathbb{F}_p^\times as an auxiliary group.

The immediate and important application of this connection is that it gives us the tightest estimate known so far on the lower bound on ECDHP for those SECG curves. We present our DLP to DHP reduction algorithm using implicit representations and \mathbb{F}_p^\times as auxiliary group in the following lemma:

Lemma 1. *Let \mathbb{G} be a additive cyclic group generated by $P \in \mathbb{G}$ and the order of P is a prime number p . Let $Q = xP \in \mathbb{G}$. Then, x can be computed using at most $2\log_2 p \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \left\lceil \sqrt{d} \right\rceil \right)$ group operations and by making at most $2\lceil \log_2 d \rceil$ calls to the DH-oracle. Here d is a positive divisor of $p-1$ and $\lceil \cdot \rceil$ is the greatest integer function.*

Proof. As already discussed, the proof is based on implicit representation of elements of \mathbb{F}_p using $\mathbb{H} = \mathbb{F}_p^\times$ as the auxiliary group. Recall the unknown x will be implicitly represented by $Q = xP \in \mathbb{G}$. Furthermore, \mathbb{F}_p^\times is a cyclic group with $\phi(p-1)$ generators, where ϕ is the Euler totient function. Since a random element in \mathbb{F}_p^\times is a generator with probability

$$\frac{\phi(p-1)}{p-1} > \frac{1}{6\log(\log(p-1))}$$

which is large enough(see [3]), it's easy to choose a generator of \mathbb{F}_p^\times .

Let ζ_0 be a generator of $\mathbb{H} = \mathbb{F}_p^\times$, then

$$x = \zeta_0^{i_0} \pmod{p} \quad (1)$$

for some integer i_0 such that $1 \leq i_0 \leq p-1$.

We want to compute i_0 explicitly and then x can be computed using above equation. Let $\zeta = \zeta_0^d \pmod{p}$. Since $d|(p-1)$, there exists unique cyclic subgroup, \mathbb{K} of $\mathbb{H} = \mathbb{F}_p^\times$ of order $\frac{p-1}{d}$, generated by ζ . Now as $(x^d)^{\frac{p-1}{d}} = 1$, it implies that $x^d \in \mathbb{K}$. Therefore, there exists unique non-negative integer j with $1 \leq j \leq \left(\frac{p-1}{d}\right)$ such that

$$x^d = \zeta^j \pmod{p} \quad (2)$$

Let $d_1 = \left\lceil \sqrt{\frac{p-1}{d}} \right\rceil$. Since j is between 1 and $\frac{p-1}{d}$, there exist unique non-negative integers u_1, v_1 with $0 \leq u_1, v_1 \leq d_1$ such that $j = u_1 d_1 - v_1$. Plugging this value of j in Equation 2, we get

$$x^d = \zeta^{u_1 d_1} \zeta^{-v_1} \pmod{p}$$

which implies,

$$\zeta^{v_1} x^d = (\zeta^{d_1})^{u_1} \pmod{p} \quad (3)$$

Recall that equality of two field elements can also be checked on their implicitly represented elements as follows: $y = z$ (in \mathbb{F}_p^\times) is equivalent to $yP = zP$ in \mathbb{G} . Therefore, above **implicit equation** in \mathbb{F}_p^\times is equivalent to following **explicit equation** in \mathbb{G} :

$$\zeta^{v_1} (x^d P) = (\zeta^{d_1})^{u_1} P \quad (4)$$

Since x^d is multiplication by x with itself d times and we know P and xP , we can compute implicit representation $x^d P$ of x^d , by making at most $2\lceil \log_2 d \rceil$ calls to the DH-oracle, using a method similar to the double-and-add algorithm [4, Section 6.3].

Looking at Equation 4, it is clear that the elements on the left-hand side can be computed using $x^d P$ for any value of v_1 with $0 \leq v_1 \leq d_1$, by repeated addition of previous terms by ζ -times. Similarly, the elements on the right-hand side can be computed for any value of u_1 with $0 \leq u_1 \leq d_1$ using P by repeated addition of previous terms by ζ^{d_1} -times. So, we compute $\zeta^{v_1} (x^d P)$ for each v_1 with $0 \leq v_1 \leq d_1$ and store them. Then, we compare them with each of right-hand side terms (similar to Baby-Step Giant-Step (BSGS) algorithm [11]) to find a match and it yields the integer $j = u_1 d_1 - v_1$.

Note that the non-negative integer $j = u_1 d_1 - v_1$ in Equation 2 is nothing but i_0 modulo $\frac{p-1}{d}$. Now to compute i_0 modulo $(p-1)$ from this integer j , we apply division algorithm on i_0 with divisor $\frac{p-1}{d}$ to get a relation between i_0 and j and it gives us, $i_0 = \left(\frac{p-1}{d}\right)t + j$ for some non-negative integer t . Observe that $0 \leq t < d$, otherwise $i_0 \geq p-1$, a contradiction. Therefore, the integer t can be written uniquely as $t = u_2 \lceil \sqrt{d} \rceil - v_2$ for some $0 \leq u_2, v_2 \leq \lceil \sqrt{d} \rceil$, again by the division algorithm. Thus, we get the following **implicit equation** in $\mathbb{H} = \mathbb{F}_p^\times$,

$$x = \zeta_0^{i_0} = \zeta_0^{j+t\left(\frac{p-1}{d}\right)} = \zeta_0^j \zeta_0^{\left(\frac{p-1}{d}\right)(u_2 \lceil \sqrt{d} \rceil - v_2)} \quad (5)$$

which is equivalent to

$$\left(\zeta_0^{\frac{p-1}{d}}\right)^{v_2} x = \left(\zeta_0^{\left(\frac{p-1}{d}\right)\lceil \sqrt{d} \rceil}\right)^{u_2} \zeta_0^j \quad (6)$$

The last **implicit equation** in $\mathbb{H} = \mathbb{F}_p^\times$ is equivalent to the following **explicit equation** in \mathbb{G} ,

$$\left(\zeta_0^{\frac{p-1}{d}}\right)^{v_2} (xP) = \left(\zeta_0^{\left(\frac{p-1}{d}\right)\lceil \sqrt{d} \rceil}\right)^{u_2} (\zeta_0^j P) \quad (7)$$

As xP and $\zeta_0^j P$ are known, we can solve for u_2, v_2 by finding a match between two sides of the Equation 7 using BSGS algorithm. This solution for u_2, v_2 would give us,

$$i_0 = \left(\frac{p-1}{d}\right) \left(u_2 \lceil \sqrt{d} \rceil - v_2\right) + j$$

Thus, we have **explicitly** computed i_0 . Lastly, we **extract** the original discrete logarithm x from this i_0 and the relation $x = \zeta_0^{i_0}$.

It is easy to see that it takes at most $2\log_2 p \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil \right)$ group operations to find a match in Equation 4 and at most $2\log_2 p \left(\left\lceil \sqrt{d} \right\rceil \right)$ group operations to find a match in Equation 7. Therefore, we have computed the discrete logarithm x using at most $2\log_2 p \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \left\lceil \sqrt{d} \right\rceil \right)$ group operations and by making at most $2\lceil \log_2 d \rceil$ calls to the DH-oracle. This completes the proof of the lemma. \square

Remark 1. One can get rid of the factor $\log_2 p$ from the above time complexity using KKM improvement [5]. Then, above time complexity reduces to $2 \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \left\lceil \sqrt{d} \right\rceil \right)$.

Remark 2. Observe that x^d is unknown in Equation 3 because x is unknown. This makes Equation 3 an **implicit equation** in $\mathbb{H} = \mathbb{F}_p^\times$. This is exactly the place where implicit representation computation comes into play, to compute implicit representation $x^d P$ of x^d . Moreover, to compute i_0 modulo $(p-1)$ from the integer j , the idea used in our algorithm is from [3, Theorem 1] which uses the division algorithm on integers along with BSGS algorithm on implicitly represented elements. In all, we have used Pohlig-Hellman algorithm once, BSGS algorithm twice in our reduction algorithm. We call this single occurrence of Pohlig-Hellman algorithm and the use of BSGS algorithm twice in our reduction algorithm, collectively as **sub-algorithm B**.

Remark 3. Our algorithm follows the general idea of DLP to DHP reduction algorithm by Maurer and Wolf with $\mathbb{H} = \mathbb{F}_p^\times$ as the auxiliary group where the unknown x is embedded implicitly into itself, i.e. $c = x$. To the best of our knowledge, our algorithm is the **first DLP to DHP reduction algorithm** that uses $\mathbb{H} = \mathbb{F}_p^\times$ as an auxiliary group but does not use the Chinese Remainder Theorem to compute the discrete logarithm.

4 Main Results

As stated earlier, to prove computational equivalence of DLP and DHP on a group \mathbb{G} of prime order p , one needs to construct an elliptic curve over \mathbb{F}_p of smooth order. This is an exceptionally hard task. Therefore, one has to look for some alternative ways to measure the hardness of DHP. The next best thing to the computation equivalence of DLP and DHP would be to somehow estimate the minimum number of group operations required to solve DHP. That is exactly what Muzereau *et al.* [12] did for the elliptic curves groups recommended for practical implementation by SECG [14]. Their idea was to construct a DLP to DHP reduction algorithm in which total number of group operations needed in the reduction algorithm should be insignificant when compared to the cost of solving DLP. Once we have such a reduction algorithm, they proposed that ratio of the cost of DLP and the number of calls to the DH-oracle needed in the algorithm gives the minimum number of group operations any algorithm that breaks DHP would require.

4.1 Proof of Theorem 2

Proof. Since we are dealing with an elliptic curve of prime order p , we assume that the best algorithm to solve elliptic curve discrete logarithm problem will take at least \sqrt{p} group operations. First, we give the general set up of estimating the lower bound on DHP.

Let C_{DLP} , C_{DHP} denote the time complexity of solving DLP and DHP respectively. Therefore, in the view of a general DLP to DHP reduction algorithm, we get $C_{DLP} = n \cdot C_{DHP} + M$ where n is the number of calls to the DH- oracle and M is the number of group operations required in the reduction algorithm. Now, if we assume that $M \ll C_{DLP}$, then we have:

$$C_{DHP} = \frac{C_{DLP} - M}{n} \approx \frac{C_{DLP}}{n}$$

If we set $T_{DH} = \frac{C_{DLP}}{n}$, the number T_{DH} is exactly what gives the minimum number of group operations needed by any algorithm to solve DHP, assuming $M \ll C_{DLP}$. This is how Muzereau *et al.* [12] estimated the minimum number of group operations required by any algorithm that solves DHP. Of course, the aim would be to make n as small as possible to have the value of T_{DH} as large as possible.

Now, we prove the lower bound on ECDHP. In case of \mathbb{G} being an elliptic curve group of prime order p , one can take $C_{ECDLP} = \sqrt{p}$ under our assumption. Since there is a divisor d of $p - 1$ such that $d \approx \sqrt[3]{p}$, then it is easy to check that $M \leq 2 \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \left\lceil \sqrt{d} \right\rceil \right) \approx \sqrt[3]{p}$, satisfying the condition $M \ll C_{ECDLP} = \sqrt{p}$. Since $n \leq 2 \lceil \log_2 d \rceil$, we finally get,

$$T_{DH} = O \left(\frac{\sqrt{p}}{\log_2 d} \right).$$

This implies that the minimum number group operation to solve ECDHP on any elliptic curve group of prime order p by any algorithm is of the order of $O \left(\frac{\sqrt{p}}{\log_2 d} \right)$ if there exists a divisor d of $(p - 1)$ of size approximately $\sqrt[3]{p}$. This completes the proof. \square

Remark 4. If we assume that a divisor d of $p - 1$ of size approximately $\sqrt[3]{p}$ exists, then the above result shows that the cost of ECDHP is getting closer to the cost ECDLP.

Remark 5. Note that the total number of group operations, M needed in the reduction algorithms of Muzereau *et al.* [12] and Bentahar [2] was also of the same order i.e. $M \approx \sqrt[3]{p}$. This indicates the importance of such a divisor d of size approximately $\sqrt[3]{p}$ in our reduction algorithm.

Remark 6. While the above reduction algorithm (with \mathbb{F}_p^\times as an auxiliary group) as well as the previous reduction algorithms (with $\bar{E}(\mathbb{F}_p)$ as an auxiliary group) both depend on Maurer and Wolf implicit representation computation, it is clear that the **sub-algorithm B** used in our reduction algorithm and **sub-algorithm A** used in previous reduction algorithms are quite different. Moreover, our reduction algorithm uses division algorithm to compute the discrete logarithm

while using **sub-algorithm B**. On the other hand, in previous reduction algorithms (which have $\bar{E}(\mathbb{F}_p)$ as an auxiliary group), the Chinese Remainder Theorem was used to compute the discrete logarithm using **sub-algorithm A**.

4.2 Improved value of T_{DH} : Advantage of \mathbb{F}_p^\times over $\bar{E}(\mathbb{F}_p)$

The difference between *sub-algorithm A* and *sub-algorithm B* as well as the change of auxiliary group from $\bar{E}(\mathbb{F}_p)$ to \mathbb{F}_p^\times both have their implications on the number of DH-oracle calls, consequently affecting the value of T_{DH} . Since *Sub-algorithm A* used in previous reduction algorithms required several iterations of Pohlig-Hellman algorithm, one had to compute a large number of implicitly represented elements in those reduction algorithms. Therefore, a large number of DH-oracle calls were needed in the previous reduction algorithms. On the other hand, our reduction algorithm while using *sub-algorithm B* requires **only** one implicitly represented element $x^d P$ of $x^d \in \mathbb{F}_p^\times$. This element can be computed by using at most $n \leq 2[\log_2 d]$ DH-oracle calls which can further be made really small by taking small value of d .

Recall that addition operation in $\bar{E}(\mathbb{F}_p)$ requires many multiplications in \mathbb{F}_p (one multiplication in \mathbb{F}_p means one DH-oracle call to compute implicit representation) and many inversions in \mathbb{F}_p (one inversion in \mathbb{F}_p means on average $\frac{3}{2}[\log_2 p]$ calls to the DH-oracle to compute implicit representation). Thus, in terms of DH-oracle calls, computing the sum of elements in $\bar{E}(\mathbb{F}_p)$ is much more expensive than multiplying elements in \mathbb{F}_p^\times .

Since our main aim through this reduction algorithm is to increase the value of T_{DH} which is inversely proportional to number of DH-oracle calls n , it will be nice to reduce the number of DH-oracle calls as much as possible. That is exactly what our reduction algorithm does using *sub-algorithm B* and \mathbb{F}_p^\times as the auxiliary group. This shows that the advantage of our reduction algorithm over previous reduction algorithms which used *sub-algorithm A* and $\mathbb{H} = \bar{E}(\mathbb{F}_p)$ as auxiliary groups, for getting improved value of T_{DH} .

4.3 Improved Lower Bound on ECDHP for SECG curves

In this section, we study about the lower bound on ECDHP for various important elliptic curves parameters [14] and show the improvement made by our reduction algorithm on the lower bound on ECDHP for those curves. These curves are recommended in SEC 2 by Standard of Efficient Cryptography Group (SECG) at Certicom Corporation to be used for practical purposes and we have been calling those curves SECG curves. These SECG curves are divided into two sub-categories: curves over prime fields of large odd characteristic and curves over binary fields. The prime p denotes the order of those SECG curves defined over prime fields of odd characteristic. For remaining SECG curves defined over binary fields, p denotes the prime divisor of the order of the curve, with a very small co-factor of either 2 or 4.

It should also be noted that SECG curves [14] include all curves recommended by NIST [13] and the most used ones in ANSI [1]. These covers the most commonly used curves in practice. Thus, these are important curves from the point of view of public key cryptography.

Muzereau *et al.* [12] used the value of T_{DH} as the lower bound on group operations to break DH-protocol and also gave the estimates for T_{DH} on various SECG curves. Thereafter, Bentahar [2] improved the previous values of T_{DH} given by Muzereau *et al.* and his estimates remain the best estimates till date.

Now, in our algorithm, with \mathbb{F}_p^\times as the auxiliary group, we have $n \leq 2\lceil \log_2 d \rceil$ and $M \leq 2 \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \lceil \sqrt{d} \rceil \right)$ where d is some divisor of $p-1$. As per the discussion above, to achieve a tighter (larger) value of T_{DH} using our reduction algorithm, one should try to make $n \leq 2\lceil \log_2 d \rceil$ as small as possible, which forces d to be small as well. On the other hand, we have to make sure that $M \approx \sqrt[3]{p}$, so that it does not violate $M \ll C_{ECDLP} = \sqrt{p}$. It is not hard to see that for really small value of d , M is inversely proportional to d . Therefore, too small value of d must not be used to avoid the violation of $M \ll C_{ECDLP} = \sqrt{p}$. Also note that $d \approx \sqrt[3]{p}$ yields $M \approx \sqrt[3]{p}$ in our reduction algorithm.

Keeping all these in mind, we factored $p-1$ and found that most of SECG curves contain divisors d which are between $\sqrt[3]{p}$ and \sqrt{p} and we have taken the smallest such d in the range $\sqrt[3]{p}$ and \sqrt{p} to compute the values in Table 1 and 2 given below. For those curves where such a divisor d does not exist, we have chosen the largest d less than $\sqrt[3]{p}$ to compute the values in the tables.

For those choices of d , we calculated *exact* number of the DH-oracle calls, $n \leq 2\lceil \log_2 d \rceil$ using binary expansion of d . The values of n thus achieved are significantly small as compared with the values of n shown by Bentahar [2] (and much smaller than those in the work of Muzereau *et al.* [12]). Consequently, these significantly small values of n resulted in much tighter (larger) values of T_{DH} for all SECG curves. Therefore, it implies that we have given the **tightest lower bound**, known so far, on ECDHP for all SECG curves [14] (except SECP224K1). In other words, our results shows the gap between the cost of ECDHP and ECDLP to be the **least** (known so far) for these curves and it leads us one step closer towards the computational equivalence of ECDHP and ECDLP for these important curves.

One additional advantage of our algorithm is that the values of M in our algorithm are less than or of almost same order as the ones given by Bentahar [2] for most of SECG curves.

Table 1 : Summary of results for curves of large prime characteristic

	SECP Curve	$\log_2 \sqrt{ E }$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$	ADV
	SECP112R1	55.89	48.34	4.59	51.30	6.90
	SECP112R2	54.90	37.54	5.88	49.01	5.51
	SECP128R1	64.00	43.45	6.02	57.98	5.58
	SECP128R2	63.00	48.23	5.49	57.51	6.11
	SECP160K1	80.00	48.39	6.55	73.45	5.45
	SECP160R1	80.00	53.85	6.30	73.70	5.70
	SECP160R2	80.00	47.53	6.70	73.30	5.30
	SECP192K1	96.00	84.31	5.36	90.64	6.84
	SECP192R1	96.00	55.51	6.97	89.03	5.23
	SECP224R1	112.00	98.50	5.55	106.45	6.85
	SECP224K1	-	-	-	-	-
	SECP256K1	128.00	86.12	7.00	121.00	5.60
	SECP256R1	128.00	86.06	7.00	121.00	5.60
	SECP384R1	192.00	141.33	7.33	184.67	5.87
	SECP521R1	260.50	196.26	7.67	252.83	6.03

Table 1 and Table 2 present the key values, $\log_2 M$, $\log_2 n$ and $\log_2 T_{DH}$ for various SECG curves. The tables also have the value of $\log_2 \sqrt{|E|}$ which refers to the assumed minimum cost of solving DLP in that particular SECG curve E . The column under ADV shows the number of security bits gained by the values of T_{DH} in our algorithm over the previous best known values of T_{DH} given by Bentahar [2]. Moreover, the present algorithm works for the curves SECP521R1, SECT571R1, SECT571K1 as well which were out of reach in previous work due to inability to construct auxiliary elliptic curves, and Tables 1 and Table 2 give the key data for these curves as well.

It should also be remarked that the current algorithm fails for the curve SECP224K1 as there does not exist any divisor of $p - 1$ of appropriate size. Therefore, Bentahar's result still gives the tightest value of T_{DH} for this curve.

To understand the advantage gained by our result over the work of Bentahar [2], as an example we consider the security of ECDHP for SECP256R1. The best known algorithm at present to solve ECDLP on this curve takes on an average 2^{128} group operations. Now, our algorithm implies that ECDHP can not be solved in less than $2^{121.00}$ group operations, in contrast to $2^{115.40}$ group operations from the work of Bentahar [2]. This shows that there is a gain factor of $2^{5.60}$ over the previous best known result given by Bentahar for the curve SECP256R1, see Table 1. If we assume that today's computational power is incapable of performing $2^{121.00}$ group operations (which is considered to be true by many), then ECDHP on the curve SECP256R1 is secure and any cryptography protocol which rely on DHP for its security can safely be implemented on the curve SECP256R1, under the assumption above.

Table 2. Summary of results for curves of even characteristic

	SECT Curve	$\log_2 \sqrt{ E }$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$	ADV
	SECT113R1	56.00	38.06	5.67	50.33	5.73
	SECT113R2	56.00	38.17	5.76	50.25	5.65
	SECT131R1	65.00	58.75	4.46	60.54	7.24
	SECT131R2	65.00	51.57	5.43	59.57	6.27
	SECT163K1	81.00	54.56	6.36	74.64	5.64
	SECT163R1	81.00	54.69	6.36	74.64	5.64
	SECT163R2	81.00	67.16	5.56	75.45	6.45
	SECT193R1	96.00	61.74	6.76	89.25	5.45
	SECT193R2	96.00	56.08	6.99	89.01	5.21
	SECT233K1	115.50	79.89	6.77	108.73	5.73
	SECT233R1	116.00	77.72	6.92	109.08	5.58
	SECT239K1	116.00	79.70	6.87	111.63	5.63
	SECT283K1	140.50	94.51	7.15	133.35	5.65
	SECT283R1	141.00	94.61	7.18	133.82	5.62
	SECT409K1	203.50	150.09	7.44	196.07	5.87
	SECT409R1	204.00	136.70	7.66	196.34	5.64
	SECT571K1	284.50	190.46	8.08	276.41	5.71
	SECT571R1	285.00	190.77	8.15	276.85	5.65

5 Conclusion

In this paper, we have presented first ever DLP to DHP reduction algorithm on a group \mathbb{G} of prime order p using \mathbb{F}_p^\times as an auxiliary group in the implicit representation method. Earlier work used elliptic curves over \mathbb{F}_p as auxiliary groups. We also established the advantage of our reduction algorithm over previously known reduction algorithms to achieve better (increased) lower bound on the number of operations needed to solve DHP. As a consequence of our reduction algorithm, we have presented the **tightest lower bound** known so far on ECDHP for all recommended SECG curves [14] (except SECP224K1). This work is of practical significance as it provides tighter security for protocols which depend on ECDHP for their security. Moreover, it leads us towards the computational equivalence of DHP and DLP for these SECG curves since the gap between the cost of DHP and DLP has been further reduced for these curves.

Acknowledgement

I wish to thank my advisor Dr. Ayan Mahalanobis for his continuous help and excellent guidance throughout this project.

References

- [1] X9 ANSI. 62: Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). *Am. Nat'l*

Standards Inst, 1999.

- [2] Kamel Bentahar. The equivalence between the DHP and DLP for elliptic curves used in practical applications, revisited. In *Cryptography and Coding*, pages 376–391. Springer, 2005.
- [3] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology-EUROCRYPT 2006*, pages 1–11. Springer, 2006.
- [4] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2008.
- [5] Shunji Kozaki, Taketeru Kutsuma, and Kazuto Matsuo. Remarks on Cheon’s algorithms for pairing-related problems. In *Pairing-Based Cryptography-Pairing 2007*, pages 302–316. Springer, 2007.
- [6] Ueli M Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology-CRYPTO’94*, pages 271–281. Springer LNCS 839, 1994.
- [7] Ueli M Maurer and Stefan Wolf. Diffie-Hellman oracles. In *Advances in Cryptology—Crypto’96*, pages 268–282. Springer LNCS 1109, 1996.
- [8] Ueli M Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999.
- [9] Ueli M Maurer and Stefan Wolf. The Diffie-Hellman Protocol. *Designs, Codes and Cryptography*, 19:147–171, 2000.
- [10] UM Maurer and S Wolf. On the difficulty of breaking the Diffie-Hellman protocol. Technical report, Technical Report 24, Department of Computer Science, ETH Zurich, 1996.
- [11] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.
- [12] Antoine Muzereau, Nigel Smart, and Frederik Vercauteren. The equivalence between the DHP and DLP for elliptic curves used in practical applications. *LMS Journal of Computation and Mathematics*, 7:50–72, 2004.
- [13] FIPS NIST. 186.2 Digital Signature Standard (DSS). *National Institute of Standards and Technology (NIST)*, 2000.
- [14] SECG. SEC 2. : Recommended Elliptic Curve Domain Parameters. See <http://www.secg.org/>, 2000.

Appendices for “Improved Lower Bound on
DHP: Towards the Equivalence of DHP and DLP
for Important Elliptic Curves Used for
Implementation ”

Prabhat Kushwaha

November 29, 2016

Appendices

A Elliptic curve domain parameters over prime field

The following data present several SECG curves [1] which are defined over some prime field of characteristic not equal to 2 and are used for practical purposes. For these curves, prime p denotes the order of the elliptic curve group and d is the suitable divisor of $p - 1$ which is used by us for various computation in Table 1.

A.1 SECP112R1

$p = 4451685225093714776491891542548933$
 $d = 140876$

A.2 SECP112R2

$p = 1112921306273428674967732714786891$
 $d = 110852811870$

A.3 SECP128R1

$p = 340282366762482138443322565580356624661$
 $d = 9476076960994$

A.4 SECP128R2

$p = 85070591690620534603955721926813660579$
 $d = 3101689558$

A.5 SECP160K1

$p = 1461501637330902918203686915170869725397159163571$
 $d = 42918291593381467397$

A.6 SECP160R1

$p = 1461501637330902918203687197606826779884643492439$
 $d = 22167198845997443$

A.7 SECP160R2

$p = 1461501637330902918203685083571792140653176136043$
 $d = 142004808588765074419$

A.8 SECP192K1

$p = 6277101735386680763835789423061264271957123915200845512077$
 $d = 43818996$

A.9 SECP192R1

$p = 6277101735386680763835789423176059013767194773182842284081$
 $d = 9564682313913860059195669$

A.10 SECP224K1

$p = 2695994666715063979466701508701964034651032708312007454899$
 4958668279
Appropriate size of divisor d of $p - 1$ not available

A.11 SECP224R1

$p = 2695994666715063979466701508701962594045780771442439172168$
 2722368061
 $d = 533642580$

A.12 SECP256K1

$p = 1157920892373161954235709850086879078528375642790749043826$
 05163141518161494337
 $d = 65709355417112419152054124$

A.13 SECP256R1

$p = 115792089210356248762697446949407573529996955224135760$
 342422259061068512044369
 $d = 71482998987075857096374359$

A.14 SECP384R1

$p = 3940200619639447921227904010014361380507973927046544666794$
 $6905279627659399113263569398956308152294913554433653942643$
 $d = 12895580879789762060783039592702$

A.15 SECP521R1

$p = 6864797660130609714981900799081393217269435300143305409394463$
 $45918554318339765539424505774633321719753296399637136332111386476$
 $8612440380340372808892707005449$
 $d = 1898873518475180724503002533770555108536$

B Elliptic curve domain parameters over \mathbb{F}_{2^m}

The following data present several SECG curves [1] which are defined over a binary field and are used for practical purposes. For these curves, prime p is the largest divisor of the order of that particular elliptic curve group (with a very small co-factor of either 2 or 4) and d is the appropriate divisor of $p - 1$ used by us for various computation in Table 2.

B.1 SECT113R1

$p = 5192296858534827689835882578830703$
 $d = 253877289037$

B.2 SECT113R2

$p = 5192296858534827702972497909952403$
 $d = 215851796187$

B.3 SECT131R1

$p = 1361129467683753853893932755685365560653$
 $d = 23348$

B.4 SECT131R2

$p = 1361129467683753853879535043412812867983$
 $d = 485524729$

B.5 SECT163K1

$p = 5846006549323611672814741753598448348329118574063$
 $d = 33118034411893094$

B.6 SECT163R1

$p = 5846006549323611672814738465098798981304420411291$
 $d = 27744064547201903$

B.7 SECT163R2

$p = 5846006549323611672814742442876390689256843201587$
 $d = 859825042$

B.8 SECT193R1

$p = 6277101735386680763835789423269548053691575186051040197193$
 $d = 1697589986603916123127$

B.9 SECT193R2

$p = 6277101735386680763835789423314955362437298222279840143829$
 $d = 4345632155805272808276901$

B.10 SECT233K1

$p = 34508731733952818937173779311385127605709409888622521263280$
87024741343
 $d = 11064269030135607689238$

B.11 SECT233R1

$p = 6901746346790563787434755862277025555839812737345013555379$
383634485463
 $d = 443484653691663066996649$

B.12 SECT239K1

$p = 22085588309729804119791218759286481494821656132170984888$
7480219215362213
 $d = 912013207122974008798076$

B.13 SECT283K1

$p = 38853377844514581418389238136470378132848117337930613242$
95874997529815829704422603873
 $d = 19578145037471479248182334822$

B.14 SECT283R1

$p = 7770675568902916283677847627294075626569625924376904889$
109196526770044277787378692871
 $d = 34107744933314238426752172695$

B.15 SECT409K1

$p = 3305279843951242994759576540163855199142023414821406096$
42324395022880711289249191050673258457777458014096366590617
731358671
 $d = 572443222870261113609193333057890$

B.16 SECT409R1

$p = 6610559687902485989519153080327710398284046829642812192$
 $84648798304157774827374805208143723762179110965979867288366$
 567526771
 $d = 133035142307481057108300314154446543724338$

B.17 SECT571K1

$p = 1932268761508629172347675945465993672149463664853217499$
 $32861762572575957114478021226813397852270671183470671280082$
 $5351461273674974066617311929682421617092503555733685276673$
 $d = 1650836032275210526255468059063336914554249497826676631916$

B.18 SECT571R1

$p = 386453752301725834469535189093198734429892732970643499865$
 $7235251451519142289560424536143999389415773083133881121926944$
 $486246872462816813070234528288303332411393191105285703$
 $d = 2160677396588220552651437946338996605699043277407755096919$

References

- [1] SECG. SEC 2. : Recommended Elliptic Curve Domain Parameters. *See*
<http://www.secg.org/>, 2000.